

Automated Testing of Casabac Applications

This document tells how to embed Casabac into automated testing environments. Two different testing environments are discussed:

- JUnit testing
- Reproduction of http calls against a web application using Casabac

JUnit Testing is a sophisticated logical testing of functions of your application. JUnit accesses functions on a certain logical API level, calls them and observes the result. - In this documentation we will not give further explanation on JUnit testing - there is enough documentation out, including documentation on concrete Java implementations. This documentation focuses on how to access Casabac adapter objects from a JUnit environment.

The reproduction of http just means that on an http protocol level a testing system records the calls against the application system. These calls are altered and repeated - simulating one or more client re-executing the same sequence on and on.

JUnit Testing

JUnit testing environments are typically started in an own virtual machine and are not running as side part of an application server.

Consequences:

- The runtime system of Casabac - normally running as part of web application - must be explicitly started.
- The adapter objects that are normally referenced by corresponding http-requests must be created and referenced explicitly. Adapter objects are always part of the Casabac session management, as consequence sessions and subsessions need to be administered as well.

Starting the Casabac Runtime

The following main program sets up the Casabac runtime:

```
public static void main(String[] args)
{
    // starting Casabac runtime
    System.setProperty("casabac.home",
        "C:/Casabac/HTMLBasedGUI/tomcat/webapps/CasabacGUI Server/");
    Params.init("C:/Casabac/HTMLBasedGUI/tomcat/webapps/CasabacGUI Server/",
        "c:/temp/",
        null);
    CasaConfig.initSingleton();
    ...
    ...
}
```

You see...:

- A system property "casabac.home" is set to point to the directory of the Casabac web application.
- The class "Params" must be initialized with again the directory location of the web application and with the name of a directory location in which log information is written.
- The class "CasaConfig" must be initialized by calling its "initSingleton()" -method.

If you are familiar with Casabac's two different runtime modes - one with file access, one without file access- then be aware of the fact that JUnit testing can only be done using the file access mode. Reason: if not defining references to the file system then Casabac will read all resources (e.g. translation files) from the Servlet engine - which is not available when starting the Casabac runtime system from a normal main-method.

Working with Sessions, Subsessions and Adapter Objects

When a browser client or SWT client is started with a certain page then the page is reflected on server side by

- a session
- a subsession
- an adapter object

Adapter objects have to live inside session and a subsession.

Consequence: if you want to embed an adapter object inside your JUnit test environment then you have first to create the session environment for this object. The following code shows how to do this:

```

IInteractionSessionMgr iism = InteractionSessionMgrFactory.getInteractionSessionMgr();

iism.createSession("4711","Junit Tester");
iism.createNewSubsession("4711","1");
Object o = iism.findAdapterInSubsession(
    (
        "4711",
        "1",
        XYZAdapter.class.getName(),
        "", // default
        "casabacyourfirstproject" // Casabac project
    )
);
XYZAdapter xyz = (XYZAdapter)o;
xyz.setFname("XXXX");
xyz.onSave();
// check result
System.out.println("Ergebnis = " + ija.getMessageShortText() + "/" + ija.getMessageType());

```

The central interface to manage sessions and subsessions ist he "IInteractionSessionMgr" interface. Please have a look into the corresponding Java-API-documentation.

Using the interface the program above created an adapter inside a session and subsession. The adapter is used afterwards to set some properties ("setFname") and to call some functions ("onSave").

Class Loader Issues

Please remember that Casabac in typical development mode is using an own class loader instance to create instances of adapter objects. Depending from your JUnit environment this may result in the fact that the casting in the line...

```
XYZAdapter xyz = (XYZAdapter)o;
```

...will lead to an error message.

Switch off Casabac class loading by setting the attribute "useownclassloader" in the file <webapps>/casabac/config/casaconfig.xml to "true".

Reproduction of http-Calls

There are quite a lot of tools available that are able to listen on http-protocol-level and then to replay certain http sequences against a web/ application server. Casabac also provides a simple tool to do so.

What to keep in Mind (I)

If you use tools recording http sequences you will see what's going on "communication level" between the browser client and the server:

- (1) Pages are picked from the server by calls to servlet "StartCasabacPage"
- (2) Content of pages are exchanged by calls to servlet "Connector"
- (3) In addition you may see: file upload calls to servlet (a) "FileUpload", calls to servlet (b) "StartDynamicContent" and (c) "StartDynamicPage".

(2) is the most important item because this is the actual data transfer between client and server. This item represents the hidden frame communication which Casabac is based on.

(1) and (3a) and (3c) are not relevant from logical point of view - they are "just" optical stuff.

We will concentrate on item (2) as consequence. If you analyze the data stream then you will find out:

- The data stream is zipped. If your tool does not support zipping properly then switch of the zipping of the data content in casaconfig.xml (attribute: zipcontent="false", default is "true").
- The data stream is a string that represents name-value pairs of property values that are sent from the client to the server. In addition each string has a prefix that declares where the string goes to (to which adapter object) and in which session and subsession the adapter object is located.

Example: the data string might look like:

```
v1903003809072000085CASA3_1097505101230101com.casabac.demoapps.HelloWorldAdapter..
```

The prefix is structured in the following way:

- The first character ("v" in the example) indicated the type of request. There are the types "i" for "init request", "v" for "method invoke request" and "m" for "adapter synchronization request". You do not have to be aware of this for replaying sequences.
- Then there is the session id: its length is described in character at position 2 and 3. In the example the length is "19". The value starts at position 20, in the example the session id is "CASA3_1097505101230".
- Then there is the subsession-id and some other information in the prefix - which is typically not relevant for replaying.

The most crucial part of the stream is the session id: if you replay an http sequence with always the same session id then the following things will happen:

- The system may throw the error ("session was already removed") - remembering after the first run that the session-id was already removed.
- If simulating multi users then all simulated users will use the same session. Consequence: (1) the session data is corrupt and (2) the performance will not have anything to do with real life performance because Casabac serializes all calls for one session. (Casabac guarantees that there are no parallel request-threads inside one session).

What must you do as consequence? You must exchange the session id for each test run. I.e. your test tool has to create an own one (e.g. somehow timestamp based) and has to replace all occurrences of the sessionid inside the http-streams that it sends to the server.

In the example above: you have to exchange the "CASA3_1097505101230" by a different, (more or less) "unique" sessionid.

But pay attention: as you saw in the text above Casabac transfers the length of the session id first and then the session id itself. This means you...

- ...either have to not only modify the session id but also the length parameter
- ...or, much easier: just pay attention to that the session id that you internally create exactly has the same length as the one that was recorded.

What's about the subsession? The subsession of Casabac is always living in a session, as consequence the subsession id needs only to be unique inside the session. As consequence there are no problems - you do not have to overwrite the subsession id.

What to keep in Mind (II)

When recording an http sequence keep in mind to always record a whole sequence, best starting the browser from the scratch and then finally closing the browser.

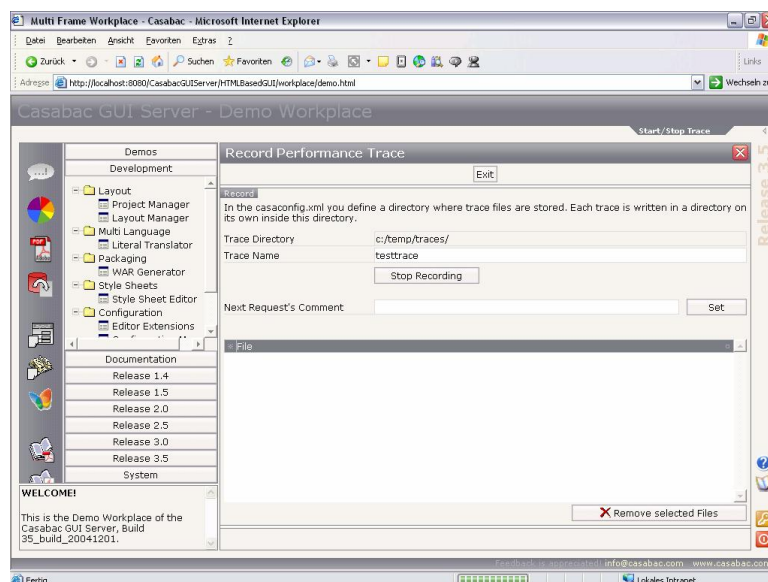
Closing the browser will send a "close session" signal to the server that will immediately remove the session and all the resources bound.

Casabac Tool for reproducing http-Sequences

As part of the Casabac toolset there is a tool for tracing and replaying requests. The tool is usable for simple test scenarios - it does for example not allow to change recorded parameters. The sequence of operations that is executed as consequence is always exactly the same.

Tracing Request

The tool is started either from the demo or from the development workplace:



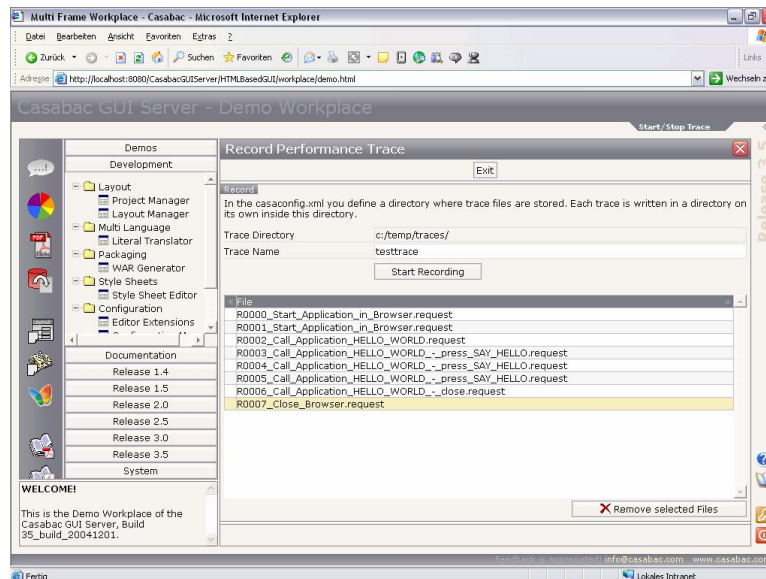
It references to a local directory in which the trace information is written. The name of this directory is defined in casaconfig.xml:

```
<casaconfig ... >
  <requestrecording recordrequests="false"
    recorddirectory="c:/temp/traces/" >
  </requestrecording >
</runtimepersistance ... >
  </runtimepersistance >
</casaconfig >
```

The tool is used in the following way:

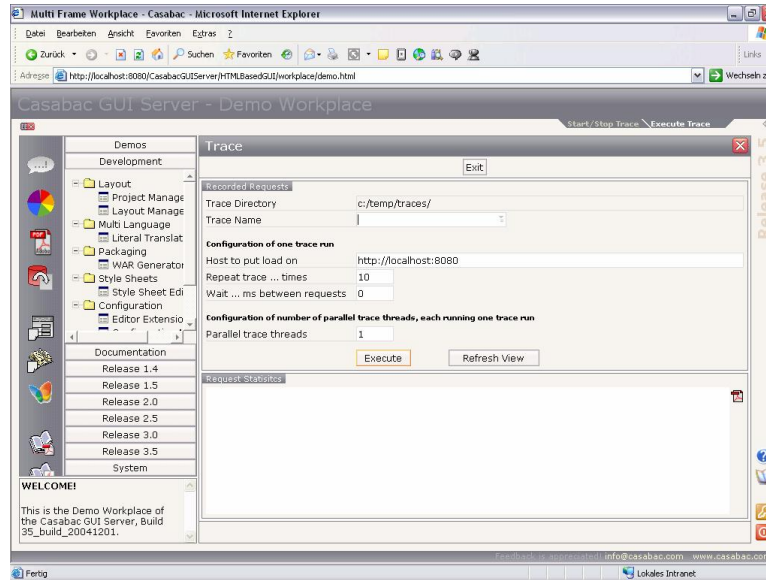
- You assign a name to your trace and start recording. Now the Casabac runtime will record any request that is not a request of the trace tool itself.
- You open a **second** browser - in this second browser you open your application using Casabac do the steps you want to trace an **close** the browser afterwards. Closing the browser is very important because this sends an http message to the server that removes the associated session!
- For each request the Casabac runtime will write one trace file. Each file gets a certain number as prefix - and a certain name. You can input the name inside the trace tool before doing the corresponding activity in the second browser.
- Stop the recording of request.

The following example shows how the trace tool looks like after recording a certain trace:



Replaying Recorded Requests

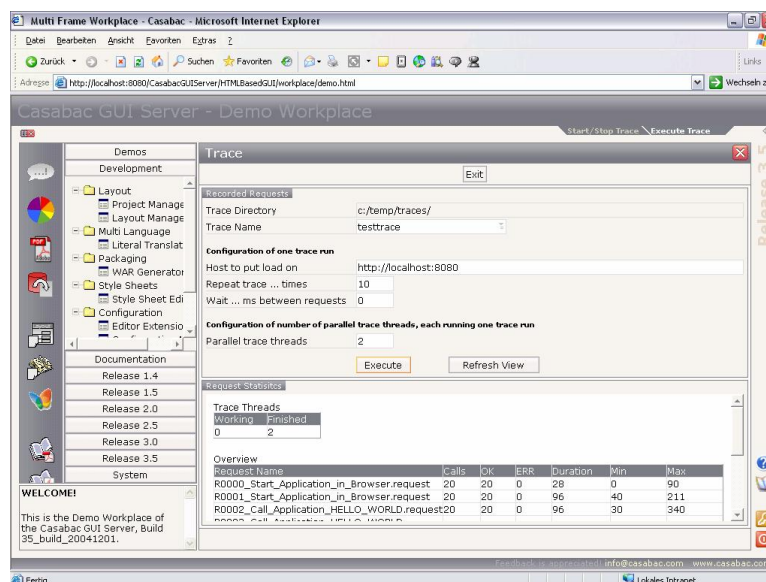
Start the second tool named "Execute Trace":



Input...

- the name of the trace
- the server you want to put load on (that of course must "understand" the request sequence recorded before)
- the number of times a sequence should be repeated and the milliseconds that should be waited for between finishing one step of the sequence and starting the next
- the number of simulated users, each one represented by a different thread, each one processing the sequence parallel to the others.

After starting, the system will start and execute the threads. By pressing "Refresh" you constantly check the status of execution. After a certain while all threads will be finished and you get a result overview:



Each request will be listed together with some statistical information. In addition there is a file in which all the trace execution information is written in detail.

Important to note: during execution "real http requests" are sent to the server that is put under load. I.e. you do not only measure the "Casabac stack" but the whole "Server stack".

Typical Usage Scenarios, Restrictions

Use the Casabac tools for recording and executing requests for...

- load analysis - simulating multiple users
- long term memory analysis - what happens with memory after executing a certain activity 1000 times?
- pressure analysis - bringing the processor of the loaded systems to 100% and see how the system behaves

There are the following restrictions:

- No http sessions are supported by the test driving client. Casabac itself does not use http sessions - but allows access to. If your application depends on this access then there will be a problem.